Introdução ao Linux:

Aula VI: sed

Prof. Dr. Marcelo Bianchi

Israel Dragone, Jamison Assunção
Leonardo Fabricius e Rafael Monteiro

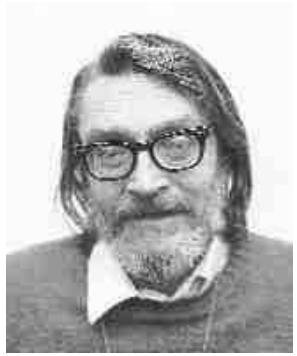
25 de julho de 2018

sed - stream editor

O que é o *sed*?

sed é um utilitário para *filtragem* e *transformações* básicas em textos, utilizando uma linguagem de programação simples e compacta.

Foi desenvolvido entre 1973 e 1974 por Lee E. McMahon da Bell Labs, e se baseou nas características do editor *ed*.



Foi uma das primeiras ferramentas com suporte às *expressões regulares*.

O que distingue o sed de outros tipos de editores é a sua habilidade de filtrar texto em um *pipeline*.

sed - stream editor

Por que usar *sed*?

Sed edita arquivos automaticamente, e desse modo, é muito eficaz caso seja necessário fazer modificações sistemáticas em muitos arquivos, ou uma modificação de um padrão que se repete muitas vezes em um arquivo muito longo.
Indo além de substituições simples, sed pode fazer alterações complexas.

Exemplos?

- Trocar uma palavra por outra em um arquivo, ou em vários arquivos (alteração simples, pode ser feita utilizando o recurso "substituir" em um editor com interface gráfica)
- Alterar o formato de uma data mm/dd/aaaa para dd/mm/aaaa (alteração complexa)

sed - stream editor

Como funciona

O programa faz uma única passagem pelo arquivo (ou stream), e aplica os comandos de forma sequencial, isto é, linha por linha.

A idéia de filtro

Um texto "exemplo.txt" é alterado, seguindo as regradas determinadas pelo "script", e a saida é um novo arquivo "exemplo-modificado.txt".

Sintaxe

Execução a partir de um arquivo:

$ sed [opções] script [arquivo]

ou uma entrada *stream* (utilizando um *pipe*)

$ cat arquivo | sed [opções] script -

O "script" tem um formato geral:

[end]X[opc]

sed - stream editor

Sintaxe dos comandos sed

[end]X[opc]

[end]: é um endereço de linha opcional, pode ser apenas um número de linha, uma expressão regular ou um intervalo de linhas

X: é um comando sed de uma letra. Se end for especificado então X é executado apenas nas referidas linhas

[opc]: opções adicionais usadas por alguns comandos.

sed - stream editor

alunos.dat

Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Anita 18 EACH Gerontologia 6

sed - stream editor

Alguns comandos e opções

Comando **p**

O comando p imprime a linha.
$ sed 'p' alunos.dat

```
Nome Idade Faculdade Curso semestre
Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
João 19 IME Computação 3
Pedro 25 ECA Música 12
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Arthur 22 POLI Naval 6
Anita 18 EACH Gerontologia 6
Anita 18 EACH Gerontologia 6
```

Neste caso, o sed, por padrão, imprime todas as linhas na saída padrão (stdout). Assim, o comando p faz com que a saída fique duplicada.

sed - stream editor

A opção **-n**

Podemos utilizar a opção -n para que o sed não imprimida nada na saída. Assim, a saída somente ocorrerá quando especificado.

```
$ sed -n 'p' alunos.dat
```

```
Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Anita 18 EACH Gerontologia 6
```

sed - stream editor

Podemos utilizar um endereço de linha para filtrar uma linha específica

```
$ sed -n '2p' alunos.dat
```

```
João 19 IME Computação 3
```

Utilizando a “,” (vírgula) é possível especificar um intervalo de linhas

```
$ sed -n '2,4p' alunos.dat
```

```
João 19 IME Computação 3
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
```

sed - stream editor

O caractere especial **$**

Como não sabemos o número de linhas que um arquivo pode ter, o caractere $ (cifrão) faz referência à última linha do texto.

Então, para imprimir da 3 linha até o final do arquivo, fazemos:

```
$ sed -n '3,$p' alunos.dat
```

```
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Anita 18 EACH Gerontologia 6
```

sed - stream editor

O comando **s** (substituição)

O comando s tem o formato geral:

's/*padrão*/*substituição*/'

Sed irá tentar encontrar o *padrão* e caso consiga "casar", isto é, encontrar uma ocorrência do *padrão* irá substituir por *substituição*.

Adicionalmente algumas opções que podem ser utilizadas no formato:

's/*padrão*/*substituição*/*opções*'

sed - stream editor

Na primeira linha do arquivo alunos.dat aparece a palavra “semestre” (com o s inicial em minúsculo).

Para manter o mesmo formato das outras colunas, com a primeira letra em maiúsculo, podemos utilizar:

$ sed 's/semestre/Semestre/' alunos.dat

```
Nome Idade Faculdade Curso Semestre
João 19 IME Computação 3
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Anita 18 EACH Gerontologia 6
```

sed - stream editor

O que acontece se houver mais de uma ocorrência da palavra procurada na linha?

```
$ echo 'linux: viva o linux' | sed 's/linux/LINUX/' -
```

```
LINUX: viva o linux
```

Neste caso, utilizamos a opção **g**, para substituir mais de uma ocorrência.

```
$ echo 'linux: viva o linux' | sed 's/linux/LINUX/g' -
```

```
LINUX: viva o LINUX
```

Separadores do comando s

No comando s foi utilizado separador /. Porém, outros caracteres podem ser utilizados.

Por exemplo:

```
$ sed 's@semestre@Semestre@' alunos.dat
```

```
$ sed 's|semestre|Semestre|' alunos.dat
```

Quando utilizar um separador diferente?

Por exemplo:

```
$ sed 's/\/usr\/local\/bin\//\/opt\/bin\//' script
```

Utilizando outro separador:

```
$ sed 's:/usr/local/bin/:/opt/bin/:' script
```

sed - stream editor

Salvando as modificações

Para salvar o arquivo modificado, ao invés de apenas exibir da tela, devemos redirecionar para um novo arquivo.

```
$ sed 's/semestre/Semestre/' alunos.dat > alunos-novo.dat
```

sed - stream editor

Salvando as modificações

Também é possível fazer as alterações e salvar no mesmo arquivo utilizando a opção -i:

```
$ sed -i 's/semestre/Semestre/' alunos.dat
```

A opção -i aceita um sufixo, e nesse caso faz uma cópia do arquivo utilizando o sufixo antes salvar as alterações:

```
$ sed -i.backup 's/semestre/Semestre/' alunos.dat
```

```
$ ls entrada*
alunos.dat   alunos.dat.backup
```

sed - stream editor

Os seguintes comandos são equivalentes:

```
$ sed 's/semestre/Semestre/' alunos.dat > saida.txt
$ sed 's/semestre/Semestre/' < alunos.dat > saida.txt
$ cat alunos.dat | sed 's/semestre/Semestre/' - > saida.txt
```

A opção **-e**

```
$ sed -e 's/semestre/Semestre/' alunos.dat > saida.txt
$ sed --expression='s/semestre/Semestre/' alunos.dat > saida.txt
```

sed - stream editor

A opção -f

O script pode ser salvo em arquivo e utilizado junto com a opção -f.

```
$ echo 's/semestre/Semestre/' > script.sed
$ cat script.sed
```

```
s/semestre/Semestre/
```

Agora, o arquivo "script.sed" pode ser utilizado com a opção -f e também é equivalente aos exemplos anteriores:

```
$ sed -f script.sed alunos.dat > saida.txt
```

```
$ sed --file=script.sed alunos.dat > saida.txt
```

sed - stream editor

O comando **q**

O sed pode ser interrompido utilizando o comando q.

Para exibir até a linha 3 e depois encerrar:
$ sed '3q' alunos.dat

```
Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
Pedro 25 ECA Música 12
```

Para exibir até a linha que contém a palavra "IME" e depois encerrar:
$ sed '/IME/q' alunos.dat

```
Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
```

sed - stream editor

O comando d

O comando d apaga a linha que casa com o endereço.

```
$ sed '2d' alunos.dat
```

```
Nome Idade Faculdade Curso semestre
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Anita 18 EACH Gerontologia 6
```

```
$ sed '/Naval/d' alunos.dat
```

```
Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
Pedro 25 ECA Música 12
Anita 18 EACH Gerontologia 6
```

sed - stream editor

Padrões no começo da linha

Para "casar" com o começo da linha, utiliza-se o caractere ^(circunflexo).

Por exemplo, para apagar as linhas que começam com A:

$ sed '/^A/d' alunos.dat

Nome Idade Faculdade Curso semestre
João 19 IME Computação 3
Pedro 25 ECA Música 12

Padrões no final da linha

De modo semelhante, o caractere $ casa com o final da linha.

Por exemplo, para incluir o caractere | no final de cada linha:

$ sed 's/$/|/' alunos.dat

```
Nome Idade Faculdade Curso semestre|
João 19 IME Computação 3|
Pedro 25 ECA Música 12|
Arthur 22 POLI Naval 6|
Anita 18 EACH Gerontologia 6|
```

sed - stream editor

O operador de negação "!"

Podemos inverter o resultado do comando utilizando o caractere ! após o endereço.

Aqui, apenas a primeira linha seria exibida se não houvesse o caractere !.
Porém, nesse caso, apenas a primeira linha NÃO é exibida.

$ sed -n '1!p' alunos.dat

João 19 IME Computação 3
Pedro 25 ECA Música 12
Arthur 22 POLI Naval 6
Anita 18 EACH Gerontologia 6

sed - stream editor

O caractere *

O caractere * é utilizado para "casar" com uma sequência de nenhuma (zero) ou mais instancias da expressão anterior (em geral, um caractere).

```
$ echo '      bom dia' | sed 's/ *\/Olá, /' -
```

```
Olá, bom dia
```

sed - stream editor

O padrão [*lista*]

Casa com qualquer caractere na *lista*.
Por exemplo, [aeiou] casa com todas as vogais.

Por exemplo, imprime as linhas que contenham “e” ou “i”:

$ sed -n '/[ei]/p' alunos.dat

Nome Idade Faculdade Curso semestre
Pedro 25 ECA Música 12
Anita 18 EACH Gerontologia 6

Variações

[^*lista*]: inverte o resultado, ou seja, casa com caracteres que não estão na lista

[*caractere1*-*caractere2*]: pode ser uma sequência, ou seja, casa com qualquer caractere entre *caractere1* e *caractere2* (por exemplo, [b-e])

sed - stream editor

Mais um comando pode ser utilizado ao mesmo tempo. Nesse caso, pode-se utilizar mais de uma vez a opção -e ou utilizar “;” (ponto-e-vírgula) como separador.

```
$ sed -n '1p; $p' alunos.dat
```

```
Nome Idade Faculdade Curso semestre
Anita 18 EACH Gerontologia 6
```

Exercícios

1. Apagar linhas em branco ou que contenham apenas espaços do arquivo bolo_cenoura.dat
2. Exibir apenas a lista de ingredientes do arquivo bolo_cenoura.dat
3. Tornar maiúscula a primeira letra de xícara(s) e colher(es) no arquivo bolo_cenoura.dat
4. Corrija as ocorrências de da palavra “que” no arquivo “texto_que.txt”.
5. Repita o exercício anterior utilizando a opção “-f”. (Utilize a opção “-e” do comando “echo”).
6. Formate o arquivo “cpf.txt” para que contenha apenas os números (um por linha).
7. Continue a formação do arquivo anterior e retire os pontos e traços dos números.
8. Converter as permissões da saída do comando 'ls -l' de simbólica para octal (ou seja, números)
9. Observe a saída do comando “history”. Remova o número no começo de cada linha (isso facilita o uso de CRTL+C / CRTL+V)

Solução dos Exercícios

1. $ sed '/^ *$/d' bolo_cenoura.dat
2. $ sed -n '/^ *[1-9]/p' bolo_cenoura.dat
3. $ sed 's/xicara/Xicara/; s/colher/Colher/' bolo_cenoura.dat
4. $ sed 's/uqe/que/; s/qeu/que/'; s/euq/que/' texto_que.txt
5. $ echo -e 's/uqe/que/\ns/qeu/que/\ns/euq/que/' > script.sed
   $ sed -f script.sed texto_que.txt
6. $ sed '/^$/d; /^[0-9]/!d' cpf.txt
7. $ sed '/^$/d; /^[0-9]/!d; s/.//g ; s/-//' cpf.txt
8. $ ls -l | sed 's/--x/1/g ; s/-w-/2/g ; s/-wx/3/g ; s/r--/4/g ; s/r-x/5/g ; s/rw-/6/g ; s/rwx/7/g ; s/--,/0/g'
9. $ history | sed 's/^[0-9]* //'